Deep graph clustering, efficiently dividing nodes into multiple disjoint clusters in an unsupervised manner, has become a crucial tool for analyzing ubiquitous graph data. Existing methods have acquired impressive clustering effects by optimizing the clustering network under the parametric condition—predefining the true number of clusters ($K_{tr}$). However, $K_{tr}$ is inaccessible in pure unsupervised scenarios, in which existing methods are incapable of inferring the number of clusters ($K$), causing limited feasibility. This article proposes the first Parameter-Agnostic Deep Graph Clustering method (PADGC), which consists of two core modules: $K$-guidence clustering and topological-hierarchical inference, to infer $K$ efficiently and gain impressive clustering predictions. Specifically, $K$-guidence clustering is employed to optimize the cluster assignments and discriminative embeddings in a mutual promotion manner under the latest updated $K$, even though $K$ may deviate from $K_{tr}$. In turn, such optimized cluster assignments are utilized to explore more accurate $K$ in the topological-hierarchical inference, which can split the dispersive clusters and merge the coupled ones. In this way, these two modules are complementarily optimized until generating the final convergent $K$ and discriminative cluster assignments. Extensive experiments on several benchmarks, including graphs and images, can demonstrate the superiority of our method. The mean values of our inferred $K$, in 11 out of 12 datasets, deviates from $K_{tr}$ by less than 1. Our method can also achieve competitive clustering effects with existing parametric deep graph clustering.

CCS Concepts: • Theory of computation → Unsupervised learning and clustering;

Additional Key Words and Phrases: Parameter-agnostic graph clustering, topological-hierarchical inference, $K$-guidence clustering, deep graph clustering.

ACM Reference format:
backbones, existing methods [4, 33, 49] always perform clustering on low-dimensional graph embeddings generated from GCNs, and optimize cluster assignment through the alignment constraint between embeddings and clusters. With the parametric condition of predefining the true number of clusters (Ktr), they have gained impressive clustering effects on various datasets. However, Ktr is inaccessible in pure unsupervised settings with utterly unknown cluster information. In such more realistic scenarios, existing methods cannot infer the number of clusters (K); thus, the final K can only be inefficiently inferred by searching K with the best performance in a wide range of K, resulting in limited practicability.

There are actually straightforward ways to deal with DGC task under unknown Ktr; that is, we first infer K on raw graph through parameter-agnostic classic methods [8, 18, 20, 54], then employ existing DGC methods to perform clustering under the inferred K. However, such K inferred from the raw features is not accurate enough, while the clustering effect would drop a lot under inaccurate K, as shown in Figure 1. Therefore, studying effective approaches for solving parameter-agnostic DGC task is necessary to be studied, which is expected to simultaneously bring out accurate K and impressive deep clustering effect. Parameter-agnostic DGC is a challenging task, and the corresponding challenges are: (1) under the condition that the initial K is extremely inaccurate, how to infer stable and accurate K and learn discriminative deep clustering assignments, (2) how to harmonically utilize both the deep graph embedding and raw graph structure to explore more accurate K.

To address the above challenges, we propose a Parameter-Agnostic Deep Graph Clustering (PADGC) method, including two core modules: K-guidence clustering and topological-hierarchical inference, that can infer K efficiently and acquire impressive clustering results. Specifically, the K-guidence clustering procedure employs mutual optimization on cluster assignments and embeddings, thus learning the discriminative assignments under the latest updated K. In turn, such learned assignments are utilized to explore more accurate K in the topological-hierarchical inference, which can split/merge the dispersive/coupled clusters according to their topological interaction density and assignment distance. In the training process, these two processes are complementarily optimized until generating the final convergent K and corresponding discriminative cluster assignments. Figure 1 demonstrates that, compared with inefficiently searching K in existing deep parametric methods, our approach can efficiently infer K close to Ktr and achieve impressive performance.

The main contributions of this work can be summarized into the following four aspects:

- We propose a parameter-agnostic graph clustering approach, that is, to the best of our knowledge, the first one to explore DGC without known Ktr.
- We learn the discriminative cluster distribution under each updated K in short-term training, thus benefiting the subsequent K inference process.
— To update $K$ being more accurate to $K_{tr}$, we design a topological-hierarchical inference method that performs splitting and merging on more dispersive and coupled clusters, respectively.
— Extensive experiments on various datasets can demonstrate that our approach can achieve comparable and even superior clustering performance than those parametric DGC ones.

2 RELATED WORK

2.1 Graph Convolutional Networks

The pioneering works [5, 43] commonly utilize recurrent neural networks or their improvements to learn node representation. Afterward, plenty of methods have been studied to make GCNs more powerful. Kipf et al. [23] introduced a linear function to the spectral filter in the degradation case of model performance, while GAT [48] trains several attention layers that learn to aggregate neighbor information of nodes. To decrease the computational complexity and enhance the practicability, SGCN [55] treats the neighbor aggregation as a pre-computing process, and only accesses the training nodes during training with low complexity. DeepGCNs [28] extend GCNs with more layers by borrowing concepts from CNNs, specifically residual/dense connections and dilated convolutions. Then, LSDAN [56] captures direct and long-distance neighbors via short-distance and long-distance attention mechanisms, while MT-GCN [59] with dual models is proposed with mutual teaching strategies to generate better predictions. [42] provides a theoretical guarantee of approximation for GNNs within constant time. Moreover, HyperTeNet [30] is designed to excavate the ternary relationships among the interacting entities sufficiently; VCHN [31] is proposed with aligning view-agnostic semantics to learn better graph representations.

2.2 Parameter-Agnostic Clustering

Describing the data distribution as a mixed Dirichlet process, the Dirichlet Process Mixture (DPM) [2] analyzes parameter-agnostic clustering problems from a Bayesian perspective. DPM models have become a common classic method while with inefficiency on large-scale data. Recently, some works [8, 18, 20, 54] have been proposed to improve the efficiency of DPM models. For example, an M-H procedure with split-merge proposals is proposed [20] that samples clusters of observations simultaneously rather than incrementally assigning observations to mixture components. And a Markov chain Monte Carlo (MCMC) sampler [8], that can be parallelized, is designed for DPM models. The MCMC sampler combines a nonergodic, restricted Gibbs iteration with split/merge proposals to produce an ergodic Markov chain, which enforces the correct stationary distribution of the Markov chain without the need for finite approximations.

There are also some recent graph-based parameter-agnostic clustering methods that utilize the constructed graph to recognize clusters of varying shapes, sizes, or densities. FINCH [41] considers

---

**Table 1. List of Abbreviations**

<table>
<thead>
<tr>
<th>Full name of a phrase</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Graph Clustering</td>
<td>DGC</td>
</tr>
<tr>
<td>Graph Convolutional Networks</td>
<td>GCNs</td>
</tr>
<tr>
<td>Parameter-Agnostic Deep Graph Clustering</td>
<td>PADGC</td>
</tr>
<tr>
<td>Dirichlet Process Mixture</td>
<td>DPM</td>
</tr>
<tr>
<td>Markov chain Monte Carlo</td>
<td>MCMC</td>
</tr>
<tr>
<td>Metropolis-Hastings</td>
<td>M-H</td>
</tr>
<tr>
<td>Topological Interaction Density</td>
<td>TID</td>
</tr>
<tr>
<td>Euclidean Distance</td>
<td>ED</td>
</tr>
<tr>
<td>Clustering Accuracy</td>
<td>ACC</td>
</tr>
<tr>
<td>Normalized Mutual Information</td>
<td>NMI</td>
</tr>
<tr>
<td>Adjusted Rand Index</td>
<td>ARI</td>
</tr>
</tbody>
</table>
that the first neighbor of each sample is all one needs to discover large chains and find the groups in the data. And by utilizing the inherent properties of the nearest neighbor graph, a density-based clustering method [29] is presented for identifying local high-density samples, while based on DBSCAN [13], RNN-DBSCAN [6] uses reverse nearest neighbor counts to estimate observation density, which can reduce complexity and improve heterogeneous density. DenMune [1] identifies dense regions using mutual nearest neighborhoods of size, besides obeying the mutual nearest neighbor consistency principle. Recently, several parameter-agnostic deep graph-based clustering approaches [38, 44, 58] have been proposed to perform clustering on the low-dimensional space. DCC [44] utilizes a nearest-neighbor graph to group points that are close in the latent space of an auto-encoder, while COMIC [38] projects raw data into one space where the projection embraces geometric and cluster assignment consistency. Aiming to use a labeled training image set belonging to a disjoint set of identities to cluster other images, LANDER [58] merges connected components predicted at each level of the hierarchy to form a new graph at the next level. Among these methods, the ones without the label information requirement can perform graph clustering by exploiting both node features and their corresponding neighbor information, which can be used as our parameter-agnostic comparison methods.

### 2.3 Parametric Deep Graph Clustering

Existing methods always perform graph clustering based on GCNs [23, 28, 48] and k-means [35]/spectral clustering [37], which mainly utilize the reconstructing or contrastive constraints to promote clustering.

The early study [47] explores the possibility of employing deep learning in graph clustering, which learns a nonlinear graph embedding through a stacked autoencoder and obtains the clustering results with the k-means algorithm [35]. Then, to make the decoder in GAE [24] learnable, MGAE [50] utilizes a single-layer autoencoder with a marginalization process for graph clustering. A powerful adversarial mutual information learning algorithm [27] is proposed to maximize the mutual information between node features and representations during the encoding process, which can obtain the most informative representation as the node features. To realize the mutual benefit between graph clustering and representation learning, DAEGC [49] generates the pseudo labels through the clustering process, so as to supervise the embedding learning. SENet [3] trains a graph neural network with the formulated objective of a continuous relaxation of the normalized min-cut problem. Aiming to alleviate the over-smoothing problem in GCNs, a smoothness sensor [21] is proposed that can adaptively terminate the current convolution once the smoothness is saturated, thus further improving clustering. SDCN [4] integrates the structural information into DGC, while two operators [36] are designed to trigger a protection mechanism and a correction mechanism against the feature randomness and drift problems, respectively. Then, by integrating the attention mechanism into DGC, AGCN and DAGC [39, 40] realize a more flexible and discriminative clustering. From the perspective of extending the applicability to various graphs, GRACE [22] provides a general solution for building suitable graph Laplacians of various graph types.

Recently, due to the superior ability to learn discriminative graph representation of graph contrastive learning, some approaches jointly optimize graph contrastive learning and clustering objectives, so as to generate better clustering results. GDCL [61] promotes graph clustering through an alignment objective and a debiased graph contrastive learning constraint, while DCRN [33] and AGC-DRR [15] utilize information correlation reduction to improve the discriminative capability of clustering results. And SCGC [34] improves clustering efficiency from the perspectives of network architecture, data augmentation in contrastive learning, and the objective function, while NCAGC [52] leverages neighborhood contrastive learning to push the embeddings
of neighbor nodes close. In addition, multi-view graph clustering has also received broad attention due to its wide application. For example, under the foundation of classic multi-view graph clustering methods: MVGL [60] and SDSNE [32], some deep works obtain superior clustering results through multi-view fusion module models [57] or neighbor information mining [12]. All these methods perform graph clustering requiring $K_{tr}$, while $K_{tr}$ is inaccessible in pure unsupervised settings. Their scalability would be limited in such more realistic scenarios due to the inability to efficiently infer $K$. In this article, we propose the first PADGC method that can efficiently infer $K$ and generate discriminative cluster assignments.

3 PARAMETER-AGNOSTIC DEEP GRAPH CLUSTERING

In this section, we will introduce the details of our proposed method, and the framework is depicted in Figure 2. As shown in Figure 2, in the $K$-guidence clustering procedure, the cluster assignments and graph embedding can promote each other under the constraint of alignment and the cluster-guided contrastive losses, thus generating the more discriminative cluster assignments. Then, we can change $K$ through the topological-hierarchical inference based on the learned discriminative cluster assignments. In such iterative optimization on cluster assignment, graph embedding, and $K$, we can obtain final inferred $K$. Finally, pseudo-supervised and cluster-guided contrastive losses are employed to jointly refine the clustering assignment under the inferred $K$.

3.1 Problem Definition

Notations. An undirected graph in the graph dataset is denoted as $G = (V, X, A)$, where $V = \{v_i\}_{i=1}^n$, and $X \in \mathbb{R}^{n \times d}$ represent the sets of $n$ nodes and node features, respectively. The feature vector of node $v_i$ is denoted as $x_i \in \mathbb{R}^d$. According to the adjacency relationships, $A \in \mathbb{R}^{n \times n}$ is the adjacent matrix of graph $G$. The basic architecture of our method includes the graph embedding network and the clustering network, denoted as $F$ and $C$, respectively. Here GCN [23] is utilized as our graph embedding network $F$ to embed the node representation:

$$H = f(X, A) = \hat{A}XW_f,$$

where $H \in \mathbb{R}^{n \times d}$ is the output embedding matrix, and $f(\cdot)$ represents the graph convolution operation in GCN. $W_f$ is the weight matrix of $f(\cdot)$. $\hat{A} = D^{-\frac{1}{2}}A D^{-\frac{1}{2}}$ is a re-normalization operation on the graph convolution process with $\hat{A} = A + I_n$, where $\hat{D}$ is the degree matrix of $\hat{A}$, and $I_n$ is the identity matrix.

Fig. 2. The pipeline of our proposed method. In iterative optimization, cluster assignments and graph embeddings can promote each other under alignment and cluster-guided contrastive losses, while $K$ is updated through topological-hierarchical inference. Finally, cluster assignments and graph embeddings are jointly optimized under the final inferred $K$. 
identity matrix. Meanwhile, the Multi Layer Perceptron is employed as our clustering network $C$ to map the node representation into the cluster assignments $Z = c(H)$, where $c(\cdot)$ is the mapping operation of $C$. And the embedding and cluster assignment of node $v_i$ are denoted as $h_i \in \mathbb{R}^d$ and $z_i \in \mathbb{R}^K$, respectively, where the value of $K$ changes dynamically with the $K$ inference process during training.

**Problem Setup.** As discussed in the above sections, existing deep parametric graph clustering methods can achieve impressive clustering effects on various datasets with the precondition of pre-defining the true number of clusters ($K_t$). But $K_t$ is inaccessible in more realistic, i.e., purely unsupervised, scenarios, in which these methods’ scalability is limited on account of their inability of inferring $K$ efficiently. Therefore, we aim to design a deep parameter-agnostic graph clustering method for efficiently inferring $K$ and gaining impressive clustering results, simultaneously, which is achieved through two core processes: $K$-guidance clustering and topological-hierarchical inference.

### 3.2 $K$-guidence Clustering

In this part, the discriminative cluster assignments, which can benefit the subsequent $K$ inference process, are learned by mutual optimization of graph embedding and cluster assignments. And this procedure is launched with an initial number of clusters.

For graph embedding learning, considering the superior ability of graph contrastive constraint in learning discriminative embeddings, we employ a cluster-guided contrastive constraint to optimize the graph embedding, which utilizes the clustering results to select the negative samples. By contrasting positive and negative pairs in $\{h_i\}_{i=1}^n$, the embeddings of positive pairs $(h_i, h_i^\top)$ are encouraged to be close, and the negative pairs $(h_i, h_{it})$ to be far from each other:

$$L_{cI} = - \sum_i \log \frac{e^{sim(h_i, h_i^\top)}}{e^{sim(h_i, h_i^\top)} + \sum_{neg} e^{sim(h_i, h_{it})}},$$

where $h_i^\top$ is the embedding of the other view of $v_i$ obtained commonly from augmentation, $sim(\cdot, \cdot)$ measures the cosine similarity of two samples. And we choose the one with a different cluster from the positive sample as the negative samples, which can alleviate the negative sampling bias [9, 61] and further improve graph embedding learning. By pulling cluster-friendly positive/negative samples closer/farther away on the graph embedding space, such a cluster-guided contrastive constraint can enforce inter-class dispersion and intra-class compactness better, thereby enhancing the discrimination of graph embeddings. In this way, more discriminative representations can be obtained under this cluster-guided contrastive learning constraint, for further promoting the clustering optimization.

To obtain the discriminative cluster assignment under $K$, we optimize the cluster assignment under the guidance of the discriminative embedding. Specifically, we first perform $k$-means with current value of $K$ on the graph embedding $H$ to obtain the pseudo cluster-assignment $Y^H = \{y_i^H\}_{i=1}^n, y_i^H \in \mathbb{R}$. According to $Y^H$, we can generate a teacher cluster soft-distribution $\hat{Y} \in \mathbb{R}^{n \times K}$ based on the graph embeddings:

$$\hat{y}_{i,k} = \frac{||h_i - \mu_k||_{L^2}}{\sum_{l=1}^K ||h_i - \mu_l||_{L^2}},$$

where $\hat{y}_{i,k}$ is the $k$th element in $\hat{y}_i$, and $\hat{y}_i$ is the teacher cluster soft-distribution of node $v_i$. $\mu_k = \sum_{i=1}^n h_i^k / n_k$ represents the center embedding of the $k$th center, while $h_i^k$ is the $i$th element in $H^k$ and $H^k \in \mathbb{R}^{n_k \times d}$ represents the embedding matrix of those nodes with $y_i^H = k$. Note that we normalize the teacher cluster soft-distribution of each node. Then, an alignment loss is utilized to
enforce the consistence between teacher cluster soft-distribution $\tilde{Y}$ and cluster distribution $Z$:

$$L_{aln} = \sum_{i=1}^{n} z_i \cdot \tilde{y}_i.$$  \hfill (4)

In the training process, the cluster-assignment and graph embeddings can benefit from each other through iterative optimization on $c(\cdot)$ and $f(\cdot)$, so that the discriminative cluster assignment can be obtained to promote the subsequent inference process on $K$.

### 3.3 Topological-Hierarchical Inference

Inspired by [8], we utilize the Metropolis-Hastings (M-H) framework [16] as the underlying framework for changing $K$ through splitting and merging, which has the spontaneous convergence property with enforcing the Markov chain’s correct stationary distribution. Based on this framework [8], we design a topological-hierarchical inference method that performs splitting and merging on more dispersive and coupled clusters, respectively, which is motivated by hierarchical clustering [53] and minimum-cut theory [11]. We first obtain the discriminative distributions after splitting or merging in all possible splitting and merging cases. Then, we select clusters with the potential to be split or merged according to the topological interaction and assignment distance between clusters. Finally, these selected clusters are integrated into the candidate set to be split or merged, and M-H framework [16] is performed on those clusters in the candidate set to determine splitting and merging decisions. Note that the current cluster distributions and the ones under all possible splitting/merging cases are called the distributions before and after splitting/merging, respectively.

Concretely, the discriminative distribution after splitting or merging is first obtained in all possible splitting and merging cases, in which the cluster before splitting/merging are already learned in Section 3.2. For the cluster distribution after merging, we can obtain it directly by merging the assignments of two clusters. However, for the sub-cluster distribution after splitting, it is apparently unreasonable to obtain it by randomly dividing the data in the master cluster. Note that for convenience, the clusters before/after splitting is represented as the master cluster and sub-cluster, respectively, in the following part. To acquire the discriminative sub-cluster distribution, a sub-clustering fully-connected layer is operated on each master cluster to generate the sub-cluster assignment $Z^{sub,k} = c^{sub}_k(H^k)$ with two sub-clusters, in which $c^{sub}_k(\cdot)$ is the operation of the sub-clustering network performing on the $k$th master cluster. Then, similar to cluster optimization with Equation (4), we also first employ $k$-means on the representation $H^k$ to obtain the pseudo sub-cluster assignment $\tilde{Y}^k \in \mathbb{R}^{n_k \times 2}$ for the $k$th master cluster:

$$\tilde{y}^k_{i,j} = \frac{\|h^k_i - \mu^k_j\|_2^2}{\epsilon^2},$$  \hfill (5)

where $\tilde{y}^k_{i,j}$ is the $i$th row and $j$th column element in $\tilde{Y}^k$. $\mu^k_j$ is the center embedding of the $j$th sub-cluster in $k$th master cluster. And the alignment loss is utilized to optimize them:

$$L_{aln}^{sub} = \sum_{k=1}^{K} \sum_{i=1}^{n_k} z^{sub,k}_i \cdot \tilde{y}^k_i,$$  \hfill (6)

where $z^{sub,k}_i$ is the $i$th element in $Z^{sub,k}$. The training process on sub-clustering layers are synchronous as the one on the clustering network, and the training processes on both sub-clustering are included in the cluster distribution optimization process, as shown in Algorithm 1.
Candidate Split/Merge Set Selection. Based on these discriminative cluster distributions in all possible splitting/merging cases, the potential clusters to be split/merged are selected according to topology and cluster distributions. As is known that the minimum-cut theory [11] aims to partition a graph into two sub-graphs, in which there is the fewest number of edges between the two sub-graphs. This implies that the fewer edges between two subgraphs, the more likely they can be split, and vice versa. Therefore, motivated by this theory, in the K inference process, the simple and direct way is to regard each cluster as a graph, and choose the candidate clusters according to the number of topological edges between graphs. Note that for convenience, the number of topological edges between clusters is called the topological interaction between clusters in the following part. However, due to the number of nodes in each cluster may be different, it is unreasonable to compare the topological interaction between different cluster pairs directly. For example, the topological interaction between two clusters with more nodes is most likely higher than the two clusters with less nodes. To avoid this problem, we define a concept called topological interaction density (TID), which measures the average number of edges between two clusters:

$$TID(G_i, G_j) = \frac{n_{edge}^{i,j}}{|G_i||G_j|}$$

(7)

where \(n_{edge}^{i,j}\) represents the number of edges that exist between \(G_i\) and \(G_j\). \(G_i\) and \(G_j\) represents the corresponding graph to the \(i\)th and \(j\)th clusters, respectively. \(|G_i|\) and \(|G_j|\) are the number of nodes in the \(i\)th and \(j\)th clusters, respectively.

Due to that a large value of TID means a high coupling between two clusters, in which the corresponding clusters are recommended for merging. Thus, for merging process, we choose the cluster pairs with the \(n_{ij}\)th highest TID. And the candidate set to be merged \(S_{meg}^{TID}\) according to this principle can be written as

$$S_{meg}^{TID} = \{(k_1, k_2) | \text{argsort}_{n_{ij}} TID(G_{k_1}, G_{k_2})\},$$

(8)

where \(G_{k_1}\) and \(G_{k_2}\) represents the sub-graphs corresponding to the \(k_1\)th and \(k_2\)th clusters. And for splitting process, considering that under the current \(K\), TID between existing separated clusters has a certain reference value. So we use the existing inter-cluster max TID as the limit to filter out over-coupled sub-clusters, and sustain remaining sub-clusters as the possible potential to-be-split ones:

$$S_{spl}^{TID} = \{k | TID(G_{sub_1}^k, G_{sub_2}^k) < TID_{max}\},$$

(9)

where \(TID_{max} = \max_{i, m=1}^K (TID(G_{i}, G_{m}))\), and \(G_{sub_1}^k\) and \(G_{sub_2}^k\) represents the sub-graphs corresponding to the two sub-clusters in \(k\)th master cluster. Although the screening condition in Equation (9) may indeed be somewhat loose, the final candidate split set is obtained under relatively stringent conditions, on account that the final candidate split set is determined based on both TID (Equation (9)) and ED (Equation (11)).

Moreover, as is known that hierarchical clustering partitions data by merging pairs of clusters with smaller distances, or splitting the two most distant sub-clusters in one master cluster. Inspired by this, we also generate the candidate sets to be split/merged according to the assignment distance between clusters. That is, those cluster pairs (sub-cluster pairs) with close (far) assignment distances should be considered likely to be merged (split). And the candidate sets to be split/merged according to this principle can be written as

$$S_{merg}^{ED} = \{(k_1, k_2) | \text{argsort}_{n_{ij}} ED(\mu_{k_1}, \mu_{k_2})\},$$

(10)

$$S_{spl}^{ED} = \{k | \text{argsort}_{n_{ij}} ED(\mu_1^k, \mu_2^k)\},$$

(11)

where \(ED(\cdot, \cdot)\) computes the Euclidean distance (ED) of the cluster embedding center.
According to these two principles, we can obtain the final candidate set to be split $S_{spl}$ or merged $S_{merg}$, which can be denoted as

$$S_{merg} = S_{merg}^{ED} \cap S_{merg}^{TID}, S_{spl} = S_{spl}^{ED} \cap S_{spl}^{TID}.$$  

(12)

Note that $n'_{if}$ and $n_{if}$ are determined according to $K$ when obtaining the splitting and merging candidate sets: $S_{spl}$ and $S_{merg}$. That is, $n'_{if}$ is equal to $K$ when $K$ is less than 4, and $n'_{if}$ is $[K/2]$ in other cases. Similarly, $n_{if}$ is determined according to $K$ when obtaining the merging candidate set $S_{merg}$. That is, $n_{if}$ is equal to $[K(K - 1)/2]$ when $K$ is less than 4, and $n_{if}$ is $[K(K - 1)/4]$ in other cases. Note that $[\cdot]$ is the floor function. After obtaining the candidate sets, the decision on changing $K$ is finally formed on them through computing Hasting ratio from the M-H framework [16]. After obtaining the latest $K$, the output dimension of the clustering layer is first updated with the latest $K$; the $K$-guidence clustering process is then performed to obtain better cluster assignments.

**Splitting and Merging Decision.** Following [8], the Hasting ratio for splitting is computed on the candidate master cluster to be split:

$$H_{split-k} = \frac{\alpha \prod_{c \in \{k_{sub1}, k_{sub2}\}} \Gamma(n_c) f_x(x_{\{c\}}; \lambda)}{\Gamma(n_k) f_x(x_{\{k\}}; \lambda)},$$

(13)

where $k_{sub1}$ and $k_{sub2}$ are the two sub-clusters of $k$th master cluster, and $n_k$ represents the number of nodes belonging to the $k$th cluster. $\Gamma(\cdot)$ is the Gamma function. $f_x(\cdot; \lambda)$ is the marginal likelihood with $\lambda$ representing the posterior hyperparameters, whose calculating details are following [7]. Thus, the split on the $k$th master cluster can be accepted if $H_{split-k} > 1$. Similarly, the Hasting ratio for merging is computed on the candidate cluster to be merged:

$$H_{merge-k_1k_2} = \frac{\Gamma(n_{k_2}) f_x(x_{\{k_2\}}; \lambda)}{\alpha \prod_{c \in \{k_1, k_2\}} \Gamma(n_c) f_x(x_{\{c\}}; \lambda)},$$

(14)

where $k_1$ and $k_2$ are the two clusters with the potential to be merged, and $\hat{k}$ represents the cluster after merging the clusters $k_1$ and $k_2$. Thus, the merge on the $k_1$th and $k_2$th clusters can be accepted.
Fig. 3. The topological-hierarchical inference process. Note that during the merging process, the clusters in the solid and the dotted circles represent the clusters before and after merging, respectively; during the splitting process, the clusters in the solid and the dotted circles represent the clusters before splitting and the sub-clusters after splitting, respectively.

if \( H_{merge-k_1k_2} > 1 \). Note that, when the merge clusters are finally determined according to Equation (14), if there is a shared cluster to be merged with two or more other clusters simultaneously, we will utilize the indicator \((TID(\cdot, \cdot) - \text{Norm}(ED(\cdot, \cdot)))\) to select the most coupled cluster with the shared cluster, and ignore the others. \( \text{Norm}(\cdot) \) represents the normalization operation. In this way, it can be guaranteed that no more than two clusters are merged simultaneously.

The topological-hierarchical inference process is demonstrated in Figure 3. As shown in Figure 3, based on the cluster distributions before and after splitting/merging obtained by clustering and sub-clustering net, we select the candidate clusters with the potential to be split or merged according to the topological interaction density \((TID(\cdot, \cdot))\) and assignment distance \(ED(\cdot, \cdot)\) between clusters. Then, the splitting and merging decisions are determined through the M-H framework on these candidate split/merge clusters.

During the training process, the discriminative cluster distribution is learned under the latest updated \( K \) in the \( K \)-guidence clustering procedure; in the inference process, the splitting and merging decisions can be decided based on these cluster distributions, so as to change \( K \). By performing these two processes iteratively, the converged \( K \) and corresponding cluster assignments can be generated.

**Refining cluster assignments under the inferred** \( K \). In the end, the final inferred \( K \) is fixed, and it is expected to promote clustering under this fixed \( K \) through the joint optimization on both the embedding and cluster assignments:

\[
\mathcal{L}_{ce} = \hat{\mathcal{L}}_{cl} - \frac{\beta}{n} \sum_{i} y_i^H \cdot \log(z_i), \tag{15}
\]

where \( y_i^H \) is the one-hot results obtained by performing \( k \)-means on \( H \). \( \hat{\mathcal{L}}_{cl} \) is the contrastive form similar to Equation (2), while to obtain a better contrastive effect by selecting more accurate negative samples, we select the ones that always belongs to different clusters on each updated \( K \) during training as negative samples. The whole algorithm can be seen in Algorithm 1.

### 4 EXPERIMENTS

In this section, we evaluate our proposed method and compare it with state-of-the-art methods on various datasets. The experimental protocol and comparison results are presented detailedly in turn.
Table 2. Statistics of Graph Benchmarks

<table>
<thead>
<tr>
<th>Citation</th>
<th>Article</th>
<th>Author</th>
<th>Movie</th>
<th>Co-purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>ACM</td>
<td>DBLP</td>
<td>IMDB</td>
<td>A-P</td>
</tr>
<tr>
<td>Nodes</td>
<td>2,708</td>
<td>3,312</td>
<td>19,717</td>
<td>169,343</td>
</tr>
<tr>
<td>Edges</td>
<td>5,469</td>
<td>4,732</td>
<td>44,338</td>
<td>1,166,243</td>
</tr>
<tr>
<td>Classes</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>40</td>
</tr>
</tbody>
</table>

Note that 'A-C', 'A-P', and 'O-A' represent the Amazon Computers, Amazon Photos, and ogbn-arxiv datasets, respectively.

4.1 Experimental Setup

Datasets. For the comprehensive comparison, we use 12 datasets that include four citation datasets (Cora, Citeseer, Pubmed, ogbn-arxiv [17]), one article network (ACM [14]), one author network (DBLP [14]), one movie network (IMDB [14]), two co-purchase networks (Amazon-Computers and Amazon-Photos [45]), and three image networks (MNIST [26], USPS [19], and STL-10 [10]). Statistics of these datasets are summarized in Table 2 below.

Citation datasets. Cora dataset contains 2,708 scientific publications divided into seven classes: Case-based Methods, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory. Each node represents a document, and node features are the bag-of-words representation of documents indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1,433 unique words. If an article cites another article, an undirected link/edge is added between them. There are 5,429 edges in total. Similar to the Cora dataset, Citeseer dataset contains 3,327 scientific publications divided into six classes: Agents, Artificial intelligence, Database, Information Retrieval, Machine Language, and Human-Computer Interaction. Node features are also bag-of-words representations with 3,037 unique words. There exist 4,732 edges between the nodes. Pubmed contains 19,717 articles divided into three classes: Journals, Reviews, and Others, while the node features have real-value entries indicating the Term Frequency-Inverse Document Frequency of the corresponding word from a dictionary. ogbn-arxiv dataset [17] represents the large citation network between all Computer Science (CS) ARXIV articles indexed by MAG [51].

Article network. For ACM dataset [14], node features represent the elements of a bag of words representing each article’s keywords, while two graphs are constructed by two types of relationships: “Co-Author” and “Co-Subject”. The total number of edges with all types in this multi-view dataset is summarized in Table 2, and so are the ones in another two multi-view datasets.

Author network. For DBLP dataset [14], node features are the elements of a bag of words representing each author’s keywords, while three graphs are derived from the relationships: “Co-Author”, “Co-Conference”, and “Co-Term”.

Movie network. For IMDB dataset [14], node features are the elements of a bag of words representing each movie, while two graphs are constructed by two types of relationships: “Co-director” and “Co-actor”.

Co-purchase Datasets. Amazon photos and Amazon computers [45] are segments of the Amazon co-purchase network dataset. Nodes represent goods, while node features are bag-of-words of product reviews, and the edges mean that two goods are purchased together.

Image Datasets. MNIST [26] includes 70,000 total grey images with 10 classes and 28×28 size, while USPS [19] includes 9,298 total grey images with 10 classes and 16×16 size. STL-10 [10] includes 5,000 RGB images with ten classes and 96×96 size. We construct a single graph for each image dataset where the structure is built between the total images.

Baselines. Due to that there are no parametric-agnostic deep graph clustering methods, the parametric-agnostic graph-based ones: COMIC [38], FINCH [41] and DenMune [1], are regarded...
as our comparison methods. And the DPM models: DPMI [46] and SASC-HDMM [18] are also regarded as the baseline. In addition, we compare our methods with several leading deep contrastive-based parametric methods: GDCL [61], DCRN [33], and NCAGC [52]. And the straightforward way, which first uses the existing more advanced parameter-agnostic method (SASC-HDMM [18]) to determine the categories, then performs graph clustering through existing deep parametric graph clustering methods (GDCL, DCRN, NCAGC) on this basis, is utilized as the comparison one.

**Implementation Details.** Our parametric-agnostic graph clustering learning basic framework includes one graph convolutional layer. Considering that $K_{tr}$ is always greater than 1, we launch the $K$-guidence clustering process by initializing $K$ to 2 in all datasets. At the initial state, the initial $K (K = 2)$ has a large deviation from $K_{tr}$; meanwhile, the parameters of the graph embedding network $F$ are not discriminative, which cannot obtain reliable cluster assignments. In this case, the negative samples selected based on such unreliable clusters in Equation (2) may even have more sampling bias than the one selected based on random sampling. Therefore, the graph embedding network $F$ is first pre-trained in few epochs under the graph contrastive constraint with the random negative sampling strategy, so as to obtain a relatively discriminative $F$ when starting to perform our method. In the $K$-guidence clustering procedure, we set the learning rate to 2e-3 in the cluster assignments optimization process, while the learning rate is set to 1e-3 in the graph embedding optimization and clustering refining processes. And the GPU-based $k$-means is used in $K$-guidence clustering procedure to ensure our training efficiency better. For those parametric-agnostic comparison methods, on the three multi-view datasets (ACM, DBLP, IMDB), we directly utilize the multiple views as the positive pairs without the additional augmentations. In addition, for a fair comparison, the embedding optimized under Equation (2) is used as the input of those comparison methods. In detail, since COMIC [38] is used for clustering multi-view data, we utilize the graph embeddings with different views as its multiple-view inputs, while the concatenation of graph embeddings with different views is regarded as the input of other methods. The averaged clustering results and corresponding standard deviation over 10 times of execution are reported.

**Evaluation Metrics.** Following MGAE [50], we use three quality metrics: Clustering Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI) to measure the clustering results. The best mapping between cluster assignment and true labels is computed using the Hungarian algorithm to measure the performance [25]. More precisely, ACC is defined as $\text{ACC} = \max_m (\sum^n_{i=1} 1(l_i = m(c_i))) / n$, where $l_i$ and $c_i$ represent the true label and predicted cluster of node $v_i$, respectively. Moreover, $\text{NMI} = \frac{I(l,c)}{\max[H(l),H(c)]}$ calculates the normalized measure of similarity between two labels of the same data, where $I(l,c)$ denotes the mutual information between the true label $l$ and predicted cluster $c$, while $H$ represents their entropy. The results of NMI do not change by permutations of clusters (classes), and they are normalized to the range of $[0, 1]$, with 0 meaning no correlation and 1 indicating perfect correlation. Another quantitative metric is the ARI, which is scaled between $-1$ and $1$. This metric computes the similarity between two clusters by considering all pairs of samples and counting pairs that are assigned to the same or different clusters in the ground truth and predicted clusters. The larger the ARI, the better the clustering performance.

### 4.2 Results Analysis

**Results on common graph datasets.** As shown in Table 3, our method is consistently comparable to existing parametric-agnostic methods on those common datasets, whose distribution of the true clusters is imbalanced. It indicates that our proposed method can effectively divide the imbalanced data into multiple groups without pre-defining $K_{tr}$. Our proposed approach is also superior
Table 3. Clustering Results on Common Graph Datasets

<table>
<thead>
<tr>
<th>Methods</th>
<th>ACC</th>
<th>NMI</th>
<th>ARI</th>
<th>ACC</th>
<th>NMI</th>
<th>ARI</th>
<th>ACC</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINCH</td>
<td>64.8±0.0</td>
<td>32.5±0.0</td>
<td>39.3±0.0</td>
<td>54.4±0.0</td>
<td>36.9±0.0</td>
<td>31.6±0.0</td>
<td>42.0±0.0</td>
<td>16.7±0.0</td>
<td>13.1±0.0</td>
</tr>
<tr>
<td>DenMune</td>
<td>66.8±0.0</td>
<td>55.5±0.0</td>
<td>46.1±0.0</td>
<td>52.9±0.0</td>
<td>34.6±0.0</td>
<td>30.8±0.0</td>
<td>49.5±0.0</td>
<td>27.2±0.0</td>
<td>20.1±0.0</td>
</tr>
<tr>
<td>COMIC</td>
<td>57.6±0.3</td>
<td>44.6±0.3</td>
<td>43.8±0.2</td>
<td>49.7±0.1</td>
<td>33.9±0.2</td>
<td>29.0±0.1</td>
<td>42.3±0.6</td>
<td>13.5±0.4</td>
<td>6.5±0.4</td>
</tr>
<tr>
<td>DPNI</td>
<td>51.5±0.7</td>
<td>40.1±0.5</td>
<td>36.6±0.4</td>
<td>45.3±0.6</td>
<td>31.0±0.6</td>
<td>25.9±0.4</td>
<td>39.8±0.4</td>
<td>17.6±0.3</td>
<td>12.9±0.5</td>
</tr>
<tr>
<td>SASC-HMM</td>
<td>65.9±0.3</td>
<td>53.8±0.3</td>
<td>44.6±0.2</td>
<td>55.7±0.7</td>
<td>39.1±0.4</td>
<td>35.8±0.2</td>
<td>52.6±0.2</td>
<td>30.2±0.2</td>
<td>15.9±0.4</td>
</tr>
<tr>
<td>SASC-HMM+GDCL</td>
<td>69.6±0.4</td>
<td>56.8±0.5</td>
<td>48.1±0.2</td>
<td>65.4±0.6</td>
<td>45.9±0.3</td>
<td>39.9±0.3</td>
<td>54.6±0.9</td>
<td>29.6±0.4</td>
<td>19.9±0.5</td>
</tr>
<tr>
<td>SASC-HMM+DCRN</td>
<td>69.0±0.7</td>
<td>58.8±0.4</td>
<td>48.5±0.5</td>
<td>64.9±0.5</td>
<td>43.0±0.5</td>
<td>60.5±0.4</td>
<td>48.5±0.4</td>
<td>27.1±0.2</td>
<td>20.5±0.3</td>
</tr>
<tr>
<td>SASC-HMM+NCAGC</td>
<td>63.8±0.5</td>
<td>53.0±0.6</td>
<td>38.9±0.3</td>
<td>60.1±0.5</td>
<td>41.9±0.4</td>
<td>55.4±0.5</td>
<td>46.9±0.5</td>
<td>27.8±0.4</td>
<td>21.8±0.6</td>
</tr>
<tr>
<td>Ours</td>
<td>75.9±0.2</td>
<td>59.5±0.3</td>
<td>55.5±0.3</td>
<td>69.4±0.6</td>
<td>45.3±0.3</td>
<td>43.9±0.4</td>
<td>67.7±0.3</td>
<td>31.5±0.6</td>
<td>29.5±0.4</td>
</tr>
</tbody>
</table>

We perform clustering on three multi-view datasets: Cora, Citeseer, and Pubmed, which also have imbalanced distributions of true clusters. As shown in Table 4, our method still shows superior clustering ability than those baselines. For example, the ACC of our approach can exceed the state-of-the-art performance on the three datasets by 3.1%, 8.1%, and 30.7%, respectively. This phenomenon shows that our proposed method can also achieve good clustering performance on imbalanced multi-view datasets in the more realistic unsupervised setting.

Results on image datasets. As shown in Table 4, on these three image datasets with balanced distributions of true clusters, the ACC of our method achieves 3.6%, 4.7%, and 9.9% relative improvements over the most competitive baselines, respectively, which indicates the broad scalability of our approach on DGC task in purely-unsupervised scenario.

Fig. 4. The visualization of cluster assignment in K inference process on the Pubmed dataset.

(b) K=2  
(c) K=4  
(d) K=3

 compared with those “straightforward” baselines, and the reasons for such a phenomenon are as follows. In those baselines, K inferred on the original features is not accurate enough; and due to the separation between the K inference and DGC processes, such an inaccurate K cannot be refined by the subsequent DGC process. Thus, the inaccurate K would result in a bad clustering effect. On the contrary, in our method, K is inferred based on both the deep discriminative embeddings and structure; meanwhile, K is constantly refined under the complementary optimization between K inference and DGC processes, generating the better K and clustering results. Figure 4 shows the visualization of the cluster assignments in the K inference process on the Pubmed dataset.

Results on multi-view graph datasets. We perform clustering on three multi-view datasets: IMDB, ACM, and DBLP, which also have imbalanced distributions of true clusters. As shown in Table 4, our method still shows superior clustering ability than those baselines. For example, the ACC of our approach can exceed the state-of-the-art performance on the three datasets by 3.1%, 8.1%, and 30.7%, respectively. This phenomenon shows that our proposed method can also achieve good clustering performance on imbalanced multi-view datasets in the more realistic unsupervised setting.

Results on image datasets. As shown in Table 4, on these three image datasets with balanced distributions of true clusters, the ACC of our method achieves 3.6%, 4.7%, and 9.9% relative improvements over the most competitive baselines, respectively, which indicates the broad scalability of our approach on DGC task in purely-unsupervised scenario.
Table 4. Clustering Results on Multi-view Graph Datasets and Image Datasets

<table>
<thead>
<tr>
<th>Methods</th>
<th>IMDB ACC</th>
<th>NMI</th>
<th>ARI</th>
<th>ACM ACC</th>
<th>NMI</th>
<th>ARI</th>
<th>DBLP ACC</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINCH</td>
<td>41.8±0.0</td>
<td>4.7±0.0</td>
<td>3.7±0.0</td>
<td>44.3±0.0</td>
<td>5.7±0.0</td>
<td>3.8±0.0</td>
<td>31.4±0.0</td>
<td>4.3±0.0</td>
<td>2.6±0.0</td>
</tr>
<tr>
<td>DenMune</td>
<td>50.9±0.0</td>
<td>5.0±0.0</td>
<td>4.4±0.0</td>
<td>87.5±0.0</td>
<td>78.1±0.0</td>
<td>80.6±0.0</td>
<td>56.7±0.0</td>
<td>54.9±0.0</td>
<td>48.4±0.0</td>
</tr>
<tr>
<td>COMIC</td>
<td>45.3±0.6</td>
<td>6.0±0.6</td>
<td>5.4±0.6</td>
<td>49.3±0.5</td>
<td>54.2±0.9</td>
<td>39.0±1.2</td>
<td>39.0±0.8</td>
<td>48.5±0.5</td>
<td>38.0±1.2</td>
</tr>
<tr>
<td>DPMI</td>
<td>42.5±0.3</td>
<td>10.9±0.5</td>
<td>5.1±0.4</td>
<td>59.0±0.4</td>
<td>63.6±0.7</td>
<td>51.4±0.6</td>
<td>59.6±0.4</td>
<td>59.9±0.4</td>
<td>50.9±0.6</td>
</tr>
<tr>
<td>SASC-HMM</td>
<td>42.1±0.6</td>
<td>15.3±0.3</td>
<td>5.4±0.6</td>
<td>68.9±0.5</td>
<td>70.2±0.5</td>
<td>60.5±0.4</td>
<td>58.5±0.5</td>
<td>60.1±0.4</td>
<td>50.4±0.4</td>
</tr>
<tr>
<td>SASC-HMM+GC</td>
<td>35.6±1.2</td>
<td>9.6±0.6</td>
<td>5.5±0.6</td>
<td>66.2±0.3</td>
<td>68.1±0.6</td>
<td>60.6±0.4</td>
<td>38.9±0.6</td>
<td>49.4±0.5</td>
<td>34.8±0.5</td>
</tr>
<tr>
<td>SASC-HMM+DCRN</td>
<td>36.2±0.9</td>
<td>7.3±0.5</td>
<td>5.0±0.4</td>
<td>70.2±0.4</td>
<td>72.5±0.5</td>
<td>69.5±0.4</td>
<td>59.5±0.4</td>
<td>61.0±0.2</td>
<td>53.6±0.4</td>
</tr>
<tr>
<td>Ours</td>
<td>33.9±0.8</td>
<td>6.8±0.9</td>
<td>4.8±0.5</td>
<td>70.6±0.4</td>
<td>72.0±0.4</td>
<td>69.6±0.7</td>
<td>59.8±0.7</td>
<td>60.9±0.6</td>
<td>53.9±0.9</td>
</tr>
</tbody>
</table>

90.5±0.8 71.0±0.3 77.4±1.1

Table 5. Comparing the Average Value of Inferred $K$ with Baselines

<table>
<thead>
<tr>
<th>Methods</th>
<th>MNIST ACC</th>
<th>NMI</th>
<th>ARI</th>
<th>USPS ACC</th>
<th>NMI</th>
<th>ARI</th>
<th>STL-10 ACC</th>
<th>NMI</th>
<th>ARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINCH</td>
<td>88.6±0.0</td>
<td>80.5±0.0</td>
<td>69.6±0.0</td>
<td>55.0±0.0</td>
<td>70.8±0.0</td>
<td>48.9±0.0</td>
<td>39.7±0.0</td>
<td>50.6±0.0</td>
<td>17.2±0.0</td>
</tr>
<tr>
<td>DenMune</td>
<td>92.3±0.0</td>
<td>86.0±0.0</td>
<td>85.3±0.0</td>
<td>77.2±0.0</td>
<td>80.3±0.0</td>
<td>84.6±0.0</td>
<td>70.6±0.0</td>
<td>73.6±0.0</td>
<td>61.4±0.0</td>
</tr>
<tr>
<td>COMIC</td>
<td>94.5±0.4</td>
<td>91.2±0.6</td>
<td>90.5±0.7</td>
<td>74.6±0.3</td>
<td>72.3±0.4</td>
<td>82.0±0.6</td>
<td>41.6±0.5</td>
<td>58.1±0.3</td>
<td>23.0±0.6</td>
</tr>
<tr>
<td>DPMI</td>
<td>87.9±0.5</td>
<td>85.6±0.4</td>
<td>68.4±0.3</td>
<td>76.6±0.2</td>
<td>78.9±0.4</td>
<td>80.5±0.4</td>
<td>61.6±0.7</td>
<td>65.4±0.5</td>
<td>55.9±0.5</td>
</tr>
<tr>
<td>SASC-HMM</td>
<td>89.7±0.2</td>
<td>86.5±0.3</td>
<td>80.9±0.6</td>
<td>75.9±0.3</td>
<td>71.6±0.6</td>
<td>79.6±0.6</td>
<td>71.2±0.4</td>
<td>74.5±0.5</td>
<td>62.1±0.5</td>
</tr>
<tr>
<td>SASC-HMM+GC</td>
<td>73.6±0.5</td>
<td>69.2±0.3</td>
<td>61.2±0.3</td>
<td>76.0±0.2</td>
<td>78.3±0.3</td>
<td>81.6±0.2</td>
<td>62.6±0.5</td>
<td>65.8±0.2</td>
<td>56.4±0.2</td>
</tr>
<tr>
<td>SASC-HMM+DCRN</td>
<td>74.9±0.5</td>
<td>71.6±0.7</td>
<td>63.9±0.7</td>
<td>75.5±0.4</td>
<td>77.6±0.4</td>
<td>80.9±0.5</td>
<td>68.0±0.6</td>
<td>71.9±0.7</td>
<td>59.4±0.4</td>
</tr>
<tr>
<td>Ours</td>
<td>79.1±0.7</td>
<td>93.9±1.2</td>
<td>95.4±0.5</td>
<td>82.9±0.3</td>
<td>86.0±0.4</td>
<td>80.0±0.7</td>
<td>81.1±0.5</td>
<td>75.7±0.3</td>
<td>71.5±0.7</td>
</tr>
</tbody>
</table>

Effect of inferring $K$. Table 5 shows that among the parameter-agnostic methods, the number of clusters obtained from our method is the closest to $K_{true}$. The reason why our method can infer better $K$ than the baselines are as follows. For those baselines, they only utilize the graph embedding to infer $K$ and ignore the rich information in the original graph structure, resulting in the inaccuracy of $K$; meanwhile, $K$ inference process is separated from the feature optimization, which cannot further refine $K$ through the discriminative graph embedding. In contrast, our method sufficiently utilizes the structure information of both optimized embeddings and original graph to explore more accurate $K$; and our $K$ inference process can be constantly refined by our learned embeddings to generate better $K$.

Ablation Study. We go on to evaluate the effectiveness of the proposed topological-hierarchical inference method and the final refinement process on the common graph and image datasets, which are presented in Table 6. We can find that our proposed candidate selection process (Equation (12)) can improve the inference accuracy of $K$, such as 35.7 vs. 30.2, and 6.5 vs. 8.5 on the ogbn-arxiv and Citeseer datasets, which reflects that this constraint is necessary for the reference process $K$. In addition, the comparison between our method and the one removing the final cluster assignment refinement can reflect that, the refining procedure can generate better clustering results under the final inferred $K$ by joint improvement between the graph embedding and clustering networks.

Compared with deep parametric methods. As shown in Figures 1 and 5, we also compare our method with existing leading deep parametric graph clustering methods. It shows that the clustering effects of the deep parametric methods get worse as $K$ deviates from $K_{tr}$ in most cases. And our results are comparable with them in most cases, even in case where there are some deviations of our inferred $K$ from $K_{tr}$.
Table 6. Ablation Study

<table>
<thead>
<tr>
<th>Methods</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>A-P</th>
<th>A-C</th>
<th>O-A</th>
<th>MNIST</th>
<th>USPS</th>
<th>STL-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o THI</td>
<td>68.9(5.9)</td>
<td>63.8(6.5)</td>
<td>62.9(3.8)</td>
<td>50.5(11.4)</td>
<td>51.8(12.6)</td>
<td>32.5(30.2)</td>
<td>96.1(9.8)</td>
<td>76.3(11.1)</td>
<td>78.5(12.6)</td>
</tr>
<tr>
<td>w/o Ref.</td>
<td>75.1(6.8)</td>
<td>67.5(6.5)</td>
<td>66.3(3.0)</td>
<td>68.0(9.3)</td>
<td>51.7(9.1)</td>
<td>34.3(35.7)</td>
<td>96.4(10.0)</td>
<td>79.9(9.6)</td>
<td>79.6(8.3)</td>
</tr>
<tr>
<td>Ours</td>
<td>75.9(6.8)</td>
<td>69.4(6.5)</td>
<td>67.3(3.0)</td>
<td>63.9(9.3)</td>
<td>55.9(9.1)</td>
<td>37.7(35.7)</td>
<td>97.1(10.0)</td>
<td>82.9(9.6)</td>
<td>81.8(8.3)</td>
</tr>
</tbody>
</table>

Note that, ’w/o THI’ means that selecting the candidate cluster with potential to be split/merged (Equation (12)) is removed in the $K$ inference process. ’w/o Ref.’ represents that the final cluster assignments refinement is removed.

**Complexity Analysis.** Space complexity. Since our results show that our inferred $K$ are always in the same order of magnitude with $K_{tr}$, we directly use $K_{tr}$ to represent our inferred $K$ in the complexity analysis. The space complexity of our model is $O(d\hat{d} + K_{tr}\hat{d})$, while the space complexities of the existing deep parameter methods – GDCL, DCRN, and NCAGC are $O(d\hat{d} + K_{tr}\hat{d})$, $O(d\hat{d} + d^2 + nd + dK_{tr})$ and $O(d\hat{d} + d^2 + n^2)$, respectively. Thus, our space complexity is consistent with existing deep parametric methods (GDCL and DCRN) in the order of magnitude, and outperforms the deep approach NCAGC.
Fig. 6. (a) and (b) are the curves in a whole training process on Citeseer and Amazon Computers datasets. Each gray dashed line represents one split/merge process guided by topological-hierarchical inference, while the training process between two gray dashed lines represents one iterative optimization process in Algorithm 1. (c) reflects the effects of hyper-parameter $\beta$ on clustering effect.

**Time complexity.** The time complexity of our utilized architecture to encode the clustering results is $O(n_{edge}d + nK_{tr})$. In the optimization process of K-guidence clustering, since our method iteratively uses Equations (2) and (4) to optimize the embedding and cluster distribution, respectively, the time complexity of each epoch is expressed as $O(n_{neg}d + ndK_{tr} + nK_{tr})$. The time complexity in the optimization process of K-guidence clustering is $O(n_{edge}d + ndK_{tr} + n_{neg}d + nK_{tr})$. Meanwhile, the one consumed by the topological-hierarchical inference is $O(K_{tr}^2 + d + nK_{tr})$, which is not prominent compared with other processes. The overall time complexity of the iterative optimization is $O(n_{edge}d + ndK_{tr} + n_{neg}d + nK_{tr} + K_{tr}^2)$. The overall time complexity of the refinement process at the end is $O(n_{edge}d + ndK_{tr} + n_{neg}d + ndK_{tr})$. In addition, the time complexities of the existing deep parameter methods – GDCL, DCRN, and NCAGC are $O(n_{edge}d + ndK_{tr} + n_{neg}d + nK_{tr})$, $O(nd + d^2 + n_{edge}d + n^2 + nd)$, and $O(n_{edge}d + n^2 + n^2 + nd)$, respectively. Therefore, our time complexity is consistent with existing deep parametric methods (GDCL and DCRN) in the order of magnitude, outperforming NCAGC.

In summary, our method can infer accurate $K$ and get discriminative clustering results without sacrificing algorithm complexity.

**Visualization of Training Process.** We show the experimental training curves on the two datasets with different types, in Figure 6(a) and (b). In almost every iterative optimization, the first and second half perform embedding and cluster distribution optimization, respectively; our $K$ inference would be executed after cluster distribution optimization.

**Parameter Analysis on $\beta$.** We explore the influence of the hyper-parameters $\beta$ on the clustering effect on three datasets: Amazon Photos, Amazon Computers, and IMDB, which are depicted in Figure 6(c), which indicates that the graph clustering performance is less sensitive to the hyper-parameters $\beta$.

**Parameter Analysis on Initial $K$.** On four datasets with different properties, we further explore the influence of initial clustering quality on the final inferred $K$, clustering performance, and the training efficiency in Figure 7. Note that we utilize the number of iterations (iter) in Algorithm 1 to represent the training efficiency. As seen in Algorithm 1, the process, including embedding learning, cluster distribution optimization, and $K$ inference, is counted as an iteration. Thus, iter can represent the training period from the start to stop. Figure 7 shows that the final inferred $K$ and clustering performance are relatively stable to the initial $K$. The training efficiency is sensitive to the initial $K$; iter becomes smaller when the initial $K$ is closer to $K_{tr}$, and vice versa. Such a
Fig. 7. (a), (b), and (c) are the affects of the initial $K$ on final inferred $K$, clustering performance, and the whole iteration number.

phenomenon is reasonable in that it will reach the inference stable state faster if the initial $K$ is closer to $K_{tr}$.

Parameter Analysis on $n'_{if}$ and $n_{if}$. We introduce $a_1$ and $a_2$ to define $n'_{if}$ and $n_{if}$ as

$$n'_{if} = \begin{cases} K, & \text{if } K < 4 \& a_1 = 0 \\ \left\lfloor \frac{K}{a_1} \right\rfloor, & \text{if } K < 4 \& a_1 > 0 \\ \left\lfloor \frac{K}{a_1+1} \right\rfloor, & \text{if } K \geq 4 \end{cases}, \quad n_{if} = \begin{cases} \min\left(\left\lfloor \frac{K(K-1)}{1+a_2} \right\rfloor, \frac{K(K-1)}{2} \right), & \text{if } K < 4 \\ \left\lfloor \frac{K(K-1)}{3+a_2} \right\rfloor, & \text{if } K \geq 4 \end{cases}$$

(16)

where $\left\lfloor x \right\rfloor$ is the floor function. In this way, we can explore the impact of different $n'_{if}$ and $n_{if}$ on our model by changing $a_1$ and $a_2$. From the above Equation, when $a_1 = 1$ and $a_2 = 1$, the condition for candidate split/merge clusters is the one mentioned in Section 3. It can be seen in Figure 8 that for $n'_{if}$, the final inferred $K$, clustering performance, and training efficiency would become relatively poor when $a_1$ is small ($a_1 = 0$). It is due to that small $a_1$ represents the loose condition for selecting candidate split clusters, which cannot eliminate highly coupled sub-clusters effectively, and slows down the stationary process of our inference method. As $a_1$ becomes larger ($a_1 > 0$), the final inferred $K$ and clustering performance become powerful and relatively insensitive to the change of $a_1$, on account that the strict condition for candidate split clusters can eliminate highly coupled sub-clusters effectively. But too large $a_1$ would lead to too strict conditions for candidate split clusters, which may only sustain the most separated sub-clusters, and eliminate other relatively separated sub-clusters with the potential to be split, in one splitting process. In such a case, the required training iterations for the inference process to reach stationary would be increased, thereby reducing the training efficiency. For $n_{if}$, the impact of $a_2$ on the final inferred $K$, clustering performance, and training efficiency is similar to that of $a_2$.

5 CONCLUSION

This article proposes a PADGC method, which can efficiently infer the number of clusters $K$ and achieve an impressive clustering effect. Specifically, we employ $K$-guidence clustering to learn the discriminative cluster distribution under the latest updated $K$. And the topological-hierarchical inference is presented to explore more accurate $K$ based on the learned cluster distribution and graph structure information. Through such complementary optimization between these two processes, the final converged $K$ and corresponding discriminative results can be acquired. In summary, our proposed parameter-agnostic graph clustering approach is the first one to explore DGC without known $K_{tr}$, which pioneers the first approach on the more practical task—PADGC. Our method can become a pioneer and provide some inspiration for future PADGC research.
Fig. 8. (a), (b), and (c) are the affects of $n_{if}$ and $n'_{if}$ on final inferred $K$, clustering performance, and the number of training iterations. The solid and dotted lines represent the influence of $a_1$ and $a_2$, respectively, on the final inferred $K$, clustering performance, and training efficiency.

**Limitations and future works.** On the class-imbalanced datasets, although our method can generate relatively accurate $K$ and clustering results, there is still a little gap compared with existing parametric DGC methods. Therefore, studying the parameter-agnostic DGC method on class-imbalanced datasets is a worthwhile research orientation for future work. In addition, enhancing the model scalability on more various graph data, such as dynamic graphs, is also a research direction to be paid attention to in the future.

**REFERENCES**


ACM Transactions on Knowledge Discovery from Data, Vol. 18, No. 3, Article 66. Publication date: January 2024.


Received 28 April 2023; revised 30 July 2023; accepted 10 November 2023